ICASE

A MICROPROCESSOR BASED DISPLAY CONTROLLER

Griffith Hamlin, Jr.
Thomas L. Boardman

Report Number 77-10

May 23, 1977

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia

Operated by the

UNIVERSITIES SPACE USRA RESEARCH ASSOCIATION

A MICROPROCESSOR BASED DISPLAY CONTROLLER*

Dr. G.A. Hamlin
ICASE, NASA-Langley Research Center, Hampton, Virginia
Dr. T.L. Boardman
Computing Center, University of Colorado, Boulder, Colorado

There have long been two problems with Computer Graphic Terminals. The first involves the trade-off between dynamic motion for limited amounts of information on refresh devices and virtually unlimited but static output on storage devices. The second involves the human engineering problem of long delays of graphic image output to terminals across standard communication lines. A solution to both problems, presented herein, is a graphic terminal system comprised of local (floppy disk) storage, a microcomputer, and a CRT which can be operated simultaneously in storage and refresh modes. This system, built around a Tektronix 4014 CRT and Intel 3000 series microprocessor elements, accepts segmented images from a host computer, stores them on the floppy disk, and then accepts short control commands causing the images to be displayed in store or refresh mode with locally augmented three-dimensional translation, rotation, and scaling.

INTRODUCTION

The ability to get real time pictorial output (graphics) from digital computers has existed for nearly twenty years. Initially, such output was generated on special purpose cathode ray tube (CRT) systems and was therefore extremely expensive both in graphics hardware costs and demands on the supporting digital computing systems. The storage CRT, first available approximately ten years ago, reduced these costs with hardware that could store graphic images on the CRT itself and be operated as a standard teletype-like terminal.

The inclusion of minicomputers into the refresh type (non-storage) graphics devices has allowed them to operate as teletype-like terminals, reduced their hardware costs, and substantially reduced their demands on supporting computing systems. This has made storage and refresh graphics terminals similar in output characteristics, cost, and demand on supporting computing systems (1).

There remains, however, a major difference between storage and refresh graphics terminals. Storage devices, by their nature, can display an essentially unlimited amount of information, but this information can only be modified (erased and rewritten) approximately once each second. Refresh devices can display less than one-tenth the information that storage can but can modify it many times per second permitting smooth, dynamic motion.

This paper describes a system which utilizes the hardware characteristics of Tektronix 4014 terminals and a custom microprocessor to produce a single terminal which permits both storage and refresh graphics. Using the storage option, a virtually unlimited amount of information may be displayed statically. Using the processing capabilities and memory of the microprocessor, coupled with a local floppy disk, a limited amount of information may be displayed with dynamic translation, rotation, and scaling in two and three dimensions.

---

*This report was prepared as a result of work performed under NASA Contract No. NAS1-14101 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

SYSTEM HARDWARE REQUIREMENTS

Development of a graphics device which can display information in both storage and refresh modes, perform dynamic translation, rotation, and scaling, and be connected to a host computer as a teletype-like terminal dictates many hardware requirements.

The first set of requirements relate to the CRT itself. A device which is capable of operating in both store and refresh modes, has adequate image size and resolution, and can be written at very high speeds is necessary. The Tektronix 4014 meets these requirements. It can store an unlimited amount of information, has a 280 line per inch resolution over a fourteen by eleven inch screen, and can be written fast enough to display several hundred lines in refresh mode.

The global requirements of the microprocessor are dictated by the devices which it interconnects: the Tektronix 4014 and the host computer which is ultimately providing the information to be displayed. The maximum information transfer rate to the 4014 is 100,000 bytes per second. This transfer rate should certainly be met to maximize the information which can be displayed in refresh mode. A typical host computer, communicating with the graphics terminal as though it were a teletype-like device across an asynchronous communication line, will operate no faster than 2000 bytes (characters) per second and perhaps as slowly as thirty bytes per second.

This very low communication rate greatly influences the design of the microprocessor. It dictates that images to be refreshed must be stored in the microcomputer. Further, transformations to these images must be performed in the microcomputer to permit their display in a dynamic, real time mode. Although not immediately obvious, images to be displayed in storage mode should also be stored in the microcomputer. If this is not done, user directed elimination of a portion of the stored image will require the entire remaining stored image to be retransmitted across the slow communications line.

The resulting terminal architecture is shown in Figure 1. It is based around a bit slice microprocessor (Intel 3000 system) configured to execute two million instructions per second to meet the requirements of three dimensional image transformation in real time. In addition, it includes an asynchronous, serial communications interface to the host computer, a floppy disk capable of storing 250,000 bytes of coded image descriptions, refresh buffer memory, a multiply/divide/trigometric function unit, and a 100,000 byte per second interface to the Tektronix 4014 CRT.

As detailed below, images generated by the host computer are transmitted to the serial interface in a character-coded form, converted to a sixteen bit internal representation, and stored as display units called segments on the floppy disk. The host then transmits highly abbreviated commands to cause the segments to be moved from the disk to the refresh buffers, transformed by matrix processor programs, and displayed in either refresh or storage mode on the Tektronix 4014.

The details of the image generation and control commands available to the host, their microprocessor representation, the microprocessor hardware, and the graphics system implementation software are described in the following sections.

USER GRAPHICS COMMANDS

The microprocessor is microprogrammed to present to the graphics user a virtual graphics display processor capable of accepting the definition of a tree structured image, storing that image, and displaying any or all parts (subtrees) of the image either in refresh or storage mode on the Tektronix 4014. Images are defined to the display processor by a series of calls from a higher level language (FORTRAN or PASCAL) to a set of graphics subroutines whose definition and capabilities reflect current display technology (2). Each subroutine call represents one user command.

Adherence to this syntax is enforced by the system. An appropriate error message is issued when the syntax is violated by an illegal sequence of subroutine calls. It would be more syntactically pleasing to incorporate these graphics commands into the syntax of the programming language, but writing a preprocessor or modifying the FORTRAN or PASCAL compiler is beyond the scope and purpose of this project.

The tree structured definition of an image consists of a collection of user numbered graphical SEGMENTS. Each SEGMENT defines a sub-image and is the only unit that possesses a name (integer) and graphical attributes. A SEGMENT consists of this set of graphical attributes (described below), a 4x4 homogeneous coordinate transformation matrix, a sequence of primitive graphics commands, graphical attribute modifying commands, or CALLS to other segments. The CALL mechanism is used to generate the tree structure of the image. Since there is no conditional branch graphics command, recursive calls would produce infinite loops and are not allowed.

There are seven segment manipulating user commands. The OPEN(N) and CLOSE(N) commands specify the initiation and termination of the definition of a particular segment, N. Opening a segment which already exists causes its contents to be erased immediately after the new definition of the segment is terminated. The APPEND(N) command is used in place of the OPEN command to add information to segment N. The DELETE(N) command causes the segment itself, as well as its contents, to be deleted. The DISPLAY(N) command causes the image subtree with root at segment N to be displayed on the Tektronix 4014 screen. The BLANK(N) command causes the subimage to cease being displayed. This will not occur immediately for segments being displayed in storage mode, but will occur upon encountering the next ERASE command. The CALL(N) command causes an instance of the subimage defined by segment N to be displayed within the current segment.

Associated with each segment is a set of graphics attributes and transformations. The values of these attributes and transformations are set to initial values when the segment is created. The user may change these values at any time after he has defined and closed the segment. These values normally apply to an entire segment and are compounded with the corresponding attributes of all segments called from the segment. The attribute values are specified by the following ten attribute modifying commands:

BLINK(N,rate)--gives blink rate of segment N.
INTENS(N,value)--specifies intensity value of segment N.
LINETYPE(value)--specifies linetype (dotted, dashed, etc.) of vectors in segment N.
TRANSLATE(N,X,Y,Z)--causes the transformation matrix associated with segment N to translate the image by an amount (X,Y,Z).
ROTATE(N,axis,angle)--causes the image to be rotated about the specified axis.
SCALE(N,axis,amount)--causes the image to be scaled about the specified axis by the specified amount.
PERSPC(N,X,Y,Z)--causes the image to be displayed in perspective, with viewpoint located at (X,Y,Z).
INTMTX(N,A)--specifies a 4x4 matrix A to be the transformation matrix associated with segment N.
STORE(N)--causes segment N to be displayed on the 4014 in storage mode.
REFRESH(N)--causes segment N to be displayed on the 4014 in refresh (write-through) mode.

The user may wish to have one or more of these attributes change as the processor is displaying the segment. This may be prescribed by issuing any of the above attribute commands while segment N is still open. This causes display instructions to be generated and inserted into the segment which change the associated attribute at that point. The changes remain in effect for the rest of the segment or until again changed.

Thus the PICTURE CONTROL user commands can be used in two different ways to generate either segment header values or picture control instructions. Thus the user can,

for example, associate transformation with each called subpicture (segment) and also associate transformation with each CALL itself. Also, by using PUSH and POP, the user can optionally cause the transformations to carry over from one subpicture to another. There is no universal agreement on which of the several possible alternatives to transformation control is best. A discussion of the advantages and disadvantages of each may be found in Thomas (4).

The actual image in a segment is described by graphic primitive commands. The POSITION(X,Y,Z) command causes the CRT beam to be moved to coordinate (X,Y,Z) in a blanked mode. The DRAW(X,Y,Z) causes the beam to draw a line of the current line-type from its current position to position (X,Y,Z). The TEXT(string) command causes the text string to be displayed at the current beam position using the current symbol height. The ERASE command causes all segments in storage mode to be erased from the Tektronix 4014 screen, and only those which are now in DISPLAY mode to be redrawn.

During operation, the graphics processor maintains a CURRENT transformation matrix and a CURRENT set of attributes. This data is automatically pushed onto a stack each time a new segment is begun. The new segment attributes and transformations are compounded with the CURRENT ones to become the new CURRENT attributes and transformations. The stack is popped after a segment is finished. In addition, the PUSH and POP commands can be used by the user to push or pop the CURRENT attributes and transformations from the stack.

A final graphic primitive command, the S4014(string) command, is used only with segment zero. No other primitive graphics commands or attribute commands may be used with segment zero. This command is included for compatibility with existing 4014 software. It places the string of characters, assumed to be valid 4014 control and data characters, directly into segment zero. The attribute commands may not be used with segment zero, which is assumed to possess the default attribute values. Of course, the 4014 control characters in the S4014 command may change these attributes, but automatic multiplication of all coordinates by the current transformation matrix is suppressed when processing segment zero.

This graphics processor is basically a display processor. It does, however, have some user input commands. The XHRON(X,Y,Z) command causes the 4014 crosshair cursor to be turned on at position (X,Y,Z). The XHROFF command turns off the crosshair cursor. The ENQXHR command causes the 4014 to send to the user program the (X,Y) coordinates of the current position of the crosshair cursor. The NEQBEM command causes the 4014 to send to the user program the current (X,Y) coordinates of the CRT beam.

REVIEW OF MICROPROCESSOR ARCHITECTURE

As indicated in Figure 1, there are four hardware requirements of the display controller: communication with the host computer across a low speed serial, asynchronous line, storage of image segments on a floppy disk, application of a transformation matrix to coordinates, and communication across a 100KB parallel connection to the Tektronix 4014. In addition, four software tasks must be performed: reformatting of image descriptions from the host into an easily manipulatible form, interpretation of host control commands (display, store, replace, translate,...), disk segment management, and user input processing. In order to reduce the hardware complexity, reduce the logic package count, and provide a flexible means of implementing the software functions, a microprocessor system was developed.

The microcomputer, based around Intel 3000 series elements, is shown in Figure 2. It includes a sixteen bit arithmetic unit (eight two-bit slices), a next instruction unit capable of accessing 512 sixty-bit words, and an interrupt unit capable of handling eight external devices. The sixty-bit instruction word is divided into the following functional fields:

(4) Control of Carry Inputs and Outputs to Arithmetic Units
(7) Next Instruction Address Function
(16) Mask Input to Arithmetic Units
(7) Arithmetic Unit Function
(16) Bits Used to Control External Devices
(3) Bits Used to Select External Device Being Activated
(1) Disable Interrupts Between This and Subsequent Instructions
(1) Disable Storage of Arithmetic Unit Result
(1) Permit External Devices to Extend the CPU Cycle
(1) Branch to the Arithmetic Unit Output Value
(1) Store Interrupt Address
(1) Memory Enable
(1) Memory Write Operation

Details of the use of these fields are described in a paper on the microcomputer architecture (3).

The microprocessor architecture is designed so as to provide two functions within the display controller. First, the various software tasks are executed by the arithmetic unit simplifying implementation, debugging, and modification of instructions and algorithms. Second, non-time-critical logic functions are implemented as combinations of arithmetic unit instructions replacing logic chains in the hardware device interfaces.

The microprocessor hardware is organized to communicate with eight external device interfaces across a seventy pin bus as indicated in Figure 3. Four of those devices are implemented: the serial interface to the host, an extended arithmetic unit providing multiply, divide, and trigometric functions, a disk interface, and an interface to the Tektronix 4014. Generally these interfaces perform handshaking and data formatting (serial to parallel conversion) for the particular devices with all sequencing, data storage and manipulation functions performed by micro-instructions.

IMPLEMENTATION OF GRAPHICS COMMANDS ON MICROPROCESSOR ARCHITECTURE

The implementation of the graphics commands on the microprocessor divides itself naturally into two tasks which can logically run in parallel. The first task resembles a small compiler. It accepts user graphics commands from the host computer and generates a segmented graphic display list. Each numbered segment of the list contains graphic primitive instructions describing a sub-image. This display list is much like the object code of any programming language compiler. The second task is an interpreter for the display list which causes the image to be drawn on the Tektronix 4014 display. The interpreter can be directed by the DISPLAY user command to interpret any selected segment or segments of the display list. If one segment contains a CALL to another segment, the interpreter recursively invokes itself to interpret the called segment.

The compiler and interpreter tasks communicate by sharing common data areas, consisting of the segments of the display list, and a segment control table (SCT). The SCT contains a two word entry for each segment which describes the current status of that segment. Its format is shown below:

WAKEUP	IN-USE	DISPLAY	DRAWN ONCE	ADDRESS
NOT USED				LENGTH

The wakeup bit and in-use bit are used to lock out both the compiler and interpreter tasks from accessing the same segment at once. Both tasks compete for segment access on equal priority.

The display bit is set upon receiving a DISPLAY host command, and informs the interpreter that this segment should be displayed (interpreted). The address and length specify where to begin interpretation of the segment, and its length.

A segment may reside on disk or in main memory. The compiler task generates and modifies segments only in main memory. A segment can be arbitrarily long and is stored in main memory in chained fixed length blocks. This length is an assembly-time parameter to the system. The first two words of each segment storage block contain the length of the storage block and a pointer to the next storage block for this segment. A null pointer indicates the end of the segment.

When the compiler exhausts memory space, it calls a disk I/O subroutine to write all blocks of some segments to disk. The disk I/O will first write to disk those segments which are not specified to be displayed. After this it will write segments which have already been displayed in storage mode. Then it will store segments which are to be displayed in storage mode. If main memory space is still needed it stores segments specified to be displayed in refresh mode.

If the interpreter is called to interpret a segment which is disk resident, it first calls the disk I/O to read the entire segment into main memory. This means that main memory size limits the image complexity (segment length) that can be displayed. This design decision was made because it simplified the interpreter. The disk is fast enough to support interpretation directly except for a worst case segment containing many move and draw commands which are fast on the 4014.

A segment contains display commands in all but the first two words. Segment display commands are either segment header information or display instructions. The first seventeen words of the first storage block of each segment contain header information describing the graphic attributes of the segment. These attributes are initially set to default values when the segment is created, and are modified as graphic attribute commands are processed by the compiler. The first sixteen words of the header contain a 4x4 homogeneous coordinate transformation matrix associated with the segment. This can be used to specify rotations, translations, scalings, and perspective transformation on the segment image. Bits in the seventeenth word specify the segment's intensity, linetype (solid, dotted, etc.), symbol height (for text), and whether it is to be displayed in storage or refresh mode. Before interpreting the display instructions of a segment, the interpreter task pushes the current attributes onto a pushdown stack and compounds the attributes in the segment header with the current attributes. This compounding consists mainly of multiplying the 4x4 transformation matrix by the current transformation matrix.

The display instructions are classified into four groups: vector instructions, CALL instruction, TEXT instruction, and picture control instructions. There are five vector instructions allowing the CRT beam to be moved blanked or unblanked from its current position to any given (X,Y) coordinates on the screen. These vector instructions are generated by the compiler task from the POSITION X, POSITION Y, POSITION Z, DRAW X, and DRAW END Y user commands. The full wordsize of the microprocessor is used to specify all coordinates to sixteen bits of precision, even though the Tektronix 4014 has only twelve bit screen addressing. The extra bits serve as guard digits against roundoff error when multiplying coordinates by the 4x4 transformation matrix.

The CALL instruction recursively invokes the interpreter on the called segment. The TEXT instruction, generated by the SYMBOL user command, displays text at the current beam position.

Seven of the nine picture control commands specify a new value for the current 4x4 transformation matrix, or any one of the other six graphic attributes. During interpretation a new current transformation matrix is calculated by multiplying the existing transformation matrix with the matrix specified by the picture control

instruction. The ROTATION user command causes the interpreter to calculate a 4x4 rotation matrix associated with the instruction. The MATRIX CONTROL user command directly specifies the matrix elements associated with the instruction.

The HALT, INTENSITY, LINETYPE, BLINKRATE, SYMBOL HEIGHT, or BEAM TYPE user commands generate picture control instructions which cause the interpreter to compound the attribute in the instruction with the corresponding current attribute. The PUSH and POP user commands cause the current attributes to be pushed or popped on the same stack used by the CALL instruction. This gives the user a way to specify graphic attributes which change dynamically with a segment, whereas the segment header specifies attributes which pertain to an entire segment.

As mentioned in section IV, the microprocessor hardware was designed to be interrupt driven. The compiler task is driven by the user commands received from the host computer. The interpreter is driven by the Tektronix 4014 interrupts. A rather unusual method of obtaining fast response to interrupts is used. When an interrupt occurs, the appropriate task executes one or more collections of micro-instructions called kernels. A kernel of microcode runs with interrupts disabled, and lasts no longer than about fifty microseconds (100 instructions). The last instruction of each kernel saves the address of the next kernel to be executed in a known location, and allows any pending interrupts to occur. The maximum kernel length is determined by the frequency of interrupts from the fastest device. In our case, the disk interrupts every sixty four microseconds and requires ten micro-seconds to service, leaving at most fourteen microseconds. Kernels of fifty micro-seconds give a four microsecond safety margin. Other devices are slower and interrupt at lower priority.

Each kernel is written so that it does not require any particular values in the general purpose CPU registers or status bits. All CPU registers except the interpreter's stack pointer and register eight may be used as temporary storage within a kernel, but inter-kernel results must be saved in main memory.

This results in very fast task switching upon an interrupt. No CPU status need be saved. The hardware, upon receiving an interrupt, traps to a particular address associated with that interrupt. Two micro-instructions branch to the address in a known memory location associated with the interrupt. Thus task switching takes only two microinstruction times (800 nanoseconds).

When a task finishes, it has nothing to do until receiving its next interrupt from its driving device. At such times it puts itself to sleep by zeroing a bit in CPU register eight which is reserved for that purpose, and branching to address zero. Address zero contains a four instruction loop, executed in interruptable mode. Each two instructions of this loop check the awake bit of one of the two tasks, and branch to the address of the next kernel of the first awake task. If all tasks are asleep, the CPU idles in this loop, awaiting an interrupt. The priority of the two tasks is determined by which one is checked first and by the priorities of their driving interrupts. Thus the operating system which multiprograms between compiler and interpreter consists of this four instruction idle loop, two words at each device's interrupt address, and the last instruction of each kernel which saves an address.

CONCLUSIONS

The basic microprocessor system along with the extended arithmetic unit, floppy disk interface, serial interface to the host computer, and 2,000 sixteen bit words of memory have been built, tested, and are now operating. A number of problems occured largely due to pushing the limits of Schottky TTL logic to minimize the CPU instruction time. Though the system is now functional, this experience and the apparent requirements on the processor indicate a one microsecond instruction time microprocessor would be superior to the 500 nanosecond implementation.

The 512-word instruction space limitation and instruction branching restrictions of the Intel 3000 have caused the most serious system problems. Currently only the interpreter portion of the software has been implemented in the microprocessor, with the compiler resident in the host computer. Another bank of 512 instructions will be added to accommodate the compiler along with appropriate context switching logic. Another type of bit slice processor such as the AM 2900 or TI 74S481 would probably have simplified the instruction programming process. Unfortunately these were not available when the project was begun.

Short of these problems, the graphics system has been a clear success. A high level set of host computer calls can generate segmented images, assigning them both structure and graphic attributes in a convenient manner. Once transmitted to the microcomputer over conventional communications equipment, they can be displayed with the full compliment of two and three dimensional transformations available. An essentially unlimited amount of information can be stored on the screen, and approximately 600 vectors can be refreshed with real time transformations.

This type of graphics system supports a new class of display problem: one where the majority of the image can be static while a portion is dynamically changing. Experience is indicating that this represents a very common and useful class of problem. Outputs of simulations, computer aided design and instruction, and data reduction are a few examples of activities which benefit from this capability.

It is actually not surprising that this type of graphics system is useful. The viewer of a dynamic image cannot perceive more than several hundred changing vectors so more refresh capability is generally not useful. Those changing vectors often, however, must be placed in the context of fixed images in order to maximize the information transfer to the viewer. It is hoped that such systems, relying on available microprocessor and display technology as demonstrated in this paper, will become generally available.

REFERENCES

1 Garrett, R.E. "Preliminary Development of Purdue's Remote Interactive Design System (PRIDES)", Purdue University Publication, June 1970.

2 Newman, W.M. and Sproull, R.F. Principles of Interactive Computer Graphics, McGraw-Hill, 1973.

3 Boardman, T.L. "A Microprocessor Architecture for Digital Device Implementation", AFIPS Conference Proceedings of the National Computer Conference, June, 1977.

4 Thomas, E. ACM Graphics Languages - 1976, a publication of the Association for Computing Machinery.

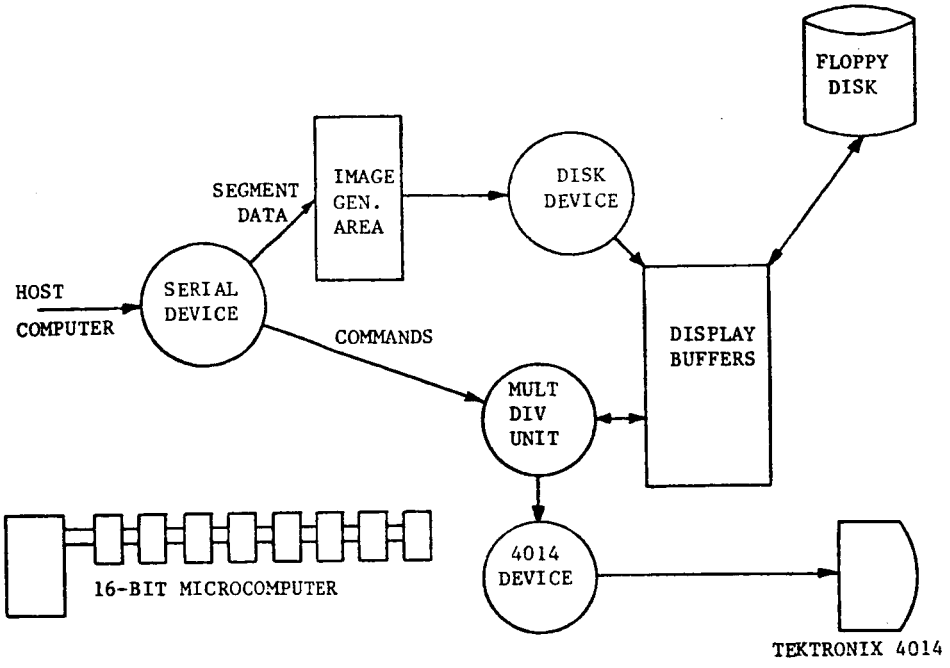

Figure 1. Microprocessor Based Display Controller

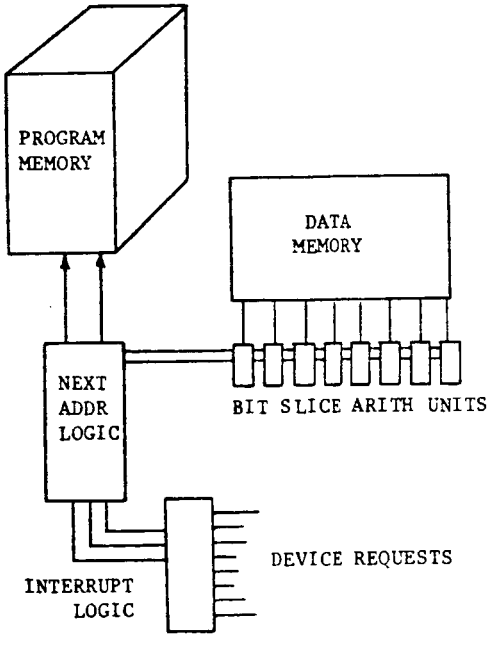

Figure 2. Microcomputer Configuration

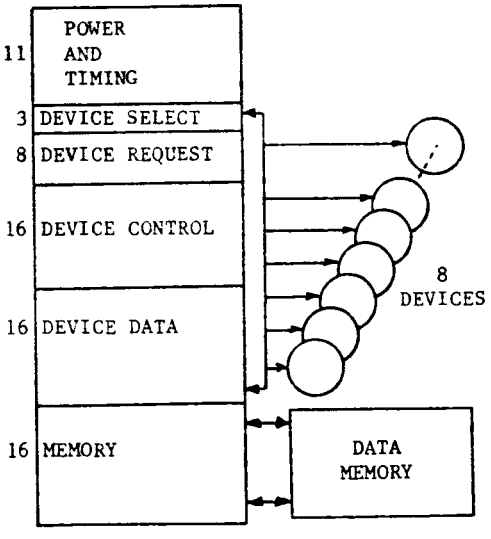

Figure 3. Bus Organization